REMARKS

By this Amendment, Applicants amend claims 3, 18, 28, and 43, add new claims 53 and 54, and amend the specification. In the final Office Action of March 7, 2006 ("Office Action") [1] claims 3-52 were rejected under 35 U.S.C. § 102(e) as anticipated by U.S. Patent No. 6,085,030 ("*Whitehead*"). In addition, the Examiner objected to the disclosure. Applicants address the rejection and objection, as well as the new claims, below.

Objection to the disclosure

The Examiner required that the cited cross reference in the specification be updated to reflect the relevant status of the cited application. By this Amendment, Applicants amend the specification to specify the patented status of the parent application (App. No. 09/030,840) and the patented status of the application (App. No. 08/636,706) mentioned on page 6 of the specification.

The Examiner also objected to the disclosure for containing an embedded hyperlink at page 7, lines 8-11. Applicants note that page 7 of the specification does not contain any hyperlinks. Applicants assume the Examiner is referring to the ftp reference on page 6, line 10. Accordingly, Applicants have removed the ftp reference on page 6, in accordance with M.P.E.P. § 608.01. Applicants therefore request the timely withdrawal of the objection to the disclosure.

Section 102(e) rejection of claims 3-52

Whitehead fails to support a § 102(e) rejection of claims 3-52. In order to properly anticipate Applicants' claimed invention under § 102, a single prior art reference must disclose

[1] The Office Action contains a number of statements reflecting characterizations of the related art and the claims. Regardless of whether any such statement is identified herein, Applicants decline to automatically subscribe to any statement or characterization in the Office Action.

each and every element of the claim at issue, either expressly or under principles of inherency. Further, "[t]he identical invention must be shown in as complete detail as is contained in the . . . claim." *See* M.P.E.P. § 2131. Also, "[t]he elements must be arranged as required by the claim." *Id.*

Independent claim 3 recites a combination including:

> sending the task request to the selected server, wherein the selected server: downloads a class definition after receiving the task request, wherein the class definition maps locations of information in the task request and allows the selected server to process the task request; extracts parameters and data from the task request using the downloaded class definition; and invokes a generic compute technique capable of executing a plurality of types of tasks, wherein the generic compute technique executes the task request using the extracted parameters and data.

Whitehead fails to disclose at least these features, as explained below.

Whitehead describes a "network component server architecture" that allows "interaction between a consumer and heterogeneous objects." Abstract; col. 1, lines 35-52. The Examiner noted *Whitehead*'s disclosure regarding an "object factory repository" and component instantiation. Office Action, pp. 3 (citing *Whitehead*, col. 10, lines 52-58, col. 11, line 65 - col. 12, line 3). The Examiner also noted *Whitehead*'s disclosure regarding downloading Java classes and applets. *Id.* (citing *Whitehead*, col. 9, lines 28-36, 42-46). Neither these cited portions nor any other portions of *Whitehead* teach the "sending" feature recited in claim 3.

Whitehead's "object factory repository" functionality does not constitute the claimed "sending." *Whitehead*'s object factory repository is "a persistent storage of implementations for the components registered in the offer repository . . . [and] provides the specific object implementation for the offered component." Col. 10, lines 49-53. In *Whitehead*'s system, component requests to the component server can be directed to a "component management service (CMS)." Col. 7, lines 20-24; col. 8, lines 3-5. If the requested component is not

available in an "offer repository 254," the CMS forwards the request to an "object factory 240," which "checks the object factory repository . . . to determine if the requested component implementation exists." Col. 8, lines 15-20. Using a repository to obtain component implementations for unregistered components, as disclosed by *Whitehead*, does not constitute "sending the task request to the selected server, wherein the selected server: downloads a class definition . . .; extracts parameters and data from the task request using the downloaded class definition; and invokes a generic compute technique capable of executing a plurality of types of tasks, wherein the generic compute technique executes the task request using the extracted parameters and data," as recited in claim 3.

Furthermore, *Whitehead*'s component instantiation functionality does not constitute the "sending" features recited in claim 3. *Whitehead* describes that the CMS supports "both local and remote instantiation of components." Col. 11, lines 55-61. According to *Whitehead*, local instantiation involves downloading and executing components on the same platform as the requesting application, while remote instantiation involves loading a local proxy object and binding the remote service to the local proxy object. Col. 11, line 61 - col. 12, line 3. Instantiating components locally and remotely, as described by *Whitehead*, does not constitute "sending the task request to the selected server, wherein the selected server: downloads a class definition . . .; extracts parameters and data from the task request using the downloaded class definition; and invokes a generic compute technique capable of executing a plurality of types of tasks, wherein the generic compute technique executes the task request using the extracted parameters and data," as recited in claim 3.

Additionally, *Whitehead*'s disclosure regarding Java classes and applets does not teach or suggest the invention defined in claim 3, including the above-noted "sending" feature.

Whitehead discloses a class loader that locates required Java component objects from a registry. The reference also mentions that the CMS is used to download Java classes and applets. While *Whitehead* mentions downloading Java classes, the reference fails to disclose downloading "a class definition after receiving the task request, wherein the class definition maps locations of information in the task request and allows the selected server to process the task request," as claimed. The reference also fails to disclose extracting parameters and data from task requests using downloaded class definitions and executing task requests using such extracted parameters and data, as recited in claim 3.

In the Office Action, the Examiner alleged that *Whitehead*'s "remote server . . . spawns a process and invokes a resident executable, i.e. a generic compute technique, for performing the requested operation." Office Action, pp. 11 (citing *Whitehead*, col. 12, lines 23-29). *Whitehead* does not support this allegation by the Examiner. The cited portion of *Whitehead* discloses that a remote machine uses a resident application to start a requested application, when the operating system does not facilitate remote start-up. Furthermore, this functionality does not constitute "sending the task request to the selected server, wherein the selected server: downloads a class definition . . .; extracts parameters and data from the task request using the downloaded class definition; and invokes a generic compute technique capable of executing a plurality of types of tasks, wherein the generic compute technique executes the task request using the extracted parameters and data," as recited in claim 3.

For at least the foregoing reasons, *Whitehead* fails to support a rejection of claim 3 under 35 U.S.C. §102(e). The § 102(e) rejection of claim 3 based on *Whitehead* should therefore be withdrawn.

Independent claim 18 recites a combination including "assembling parameters and data from the task request into a task, using the downloaded class definition" and "invoking a generic compute method, capable of processing a plurality of types of tasks, on the server, wherein the generic compute method executes the task and generates results." *Whitehead* fails to disclose at least these features.

For example, *Whitehead*'s disclosure regarding an "object factory repository" and component instantiation (col. 10, lines 52-58, col. 11, line 65 - col. 12, line 3) fails to teach the claimed "assembling" and "invoking" features. Indeed, using a repository to obtain component implementations for unregistered components, as disclosed by *Whitehead*, does not constitute "assembling parameters and data from the task request into a task, using the downloaded class definition" and "invoking a generic compute method, capable of processing a plurality of types of tasks, on the server, wherein the generic compute method executes the task and generates results," as recited in claim 18. Further, instantiating components locally and remotely does not constitute "assembling parameters and data from the task request into a task, using the downloaded class definition" and "invoking a generic compute method, capable of processing a plurality of types of tasks, on the server, wherein the generic compute method executes the task and generates results," as claimed.

In addition, *Whitehead*'s disclosure regarding Java classes and applets does not teach or suggest the recitations of claim 18, including the "assembling" and "invoking" features. While *Whitehead* might disclose downloading Java classes, the reference fails to disclose "assembling parameters and data from the task request into a task, using the downloaded class definition." Also, for at least reasons similar to those presented above in connection with claim 3, *Whitehead* fails to disclose downloading a class definition after receiving a task request, wherein the class

definition maps locations of information in the task request and allows the server to process the task request, as recited in claim 18.

For at least the foregoing reasons, *Whitehead* fails to support a rejection of claim 18 under 35 U.S.C. §102(e). The § 102(e) rejection of claim 18 should therefore be withdrawn.

Independent claim 28, although of different scope than claim 3, includes recitations similar to those of claim 3. Claim 28 is thus distinguishable from *Whitehead* for at least reasons similar to those presented above in connection with claim 3. The § 102(e) rejection of claim 28 is therefore unsupported by *Whitehead* and should be withdrawn.

Independent claim 43, although of different scope than claim 18, includes recitations similar to those of claim 18. Claim 43 is thus distinguishable from *Whitehead* for at least reasons similar to those presented above in connection with claim 18. The § 102(e) rejection of claim 43 is therefore unsupported by *Whitehead* and should be withdrawn.

Claims 4-17 depend upon claim 3; claims 19-27 depend upon claim 18; claims 29-42 depend upon claim 28; and claims 44-52 depend upon claim 43. *Whitehead* fails to support the rejection of claims 4-17, 19-27, 29-42, and 44-52 under 35 U.S.C. § 102(e) for at least reasons similar to those presented above in connection with independent claims 3, 18, 28, and 43, respectively. The § 102(e) rejection of claims 4-17, 19-27, 29-42, and 44-52 should therefore be withdrawn. Accordingly, Applicants request withdrawal of the § 102(e) rejection and the timely allowance of pending claims 3-52.

New claims 53 and 54

New claims 53 and 54 depend upon claim 3 and are distinguishable from *Whitehead* for at least reasons similar to those presented above in connection with claim 3. The applied art further fails to teach or suggest at least "[downloading] the class definition from a location indicated by a URL parameter in the task request," as recited in new claim 53. *Whitehead* also

fails to teach or suggest at least "[providing] the task request as a parameter to the generic compute technique," as recited in new claim 54. Indeed, the reference fails to teach or suggest invoking a generic compute technique as claimed, let alone "[providing] the task request as a parameter to the generic compute technique," as recited in new claim 54.

For at least the foregoing reasons, *Whitehead* fails to anticipate or render obvious new claims 53 and 54. Applicants therefore request the timely allowance of these claims.

Conclusion

Applicants request the reconsideration and reexamination of this application in view of the foregoing and the timely allowance of the pending claims.

Please grant any extensions of time required to enter this response and charge any additional required fees to our deposit account 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: August 7, 2006

By: _____
Frank A. Italiano
Reg. No. 53,056